Creating Dataextracts with DATAEXTRACT

Creating Dataextracts with DATAEXTRACT

With Adabas Load, extracts from the database can be provided in external files.

In contrast to the functions TABLEEXTRACT and TABLEUNLOAD, the user can determine the format of the target file and further process the created file.

Files created with the DATAEXTRACT statement can easily be reloaded with a corresponding DATALOAD statement.

This chapter covers the following topics:

- The DATAEXTRACT Statement
- Format Specifications for the Output File
- Structure of the Target File
- Data Formats in the Target File
- Options for the Output of Numerical Columns
- Output of NULL Values
- Text Constants in the Target File
- Generating Command Files with DATAEXTRACT
- Test Output on Screen

The DATAEXTRACT Statement

The DATAEXTRACT statement consists of a database query and the OUTFILE description.

Example 1:

```
DATAEXTRACT
    firstname, name, city FROM customer;
OUTFILE cmaster.data
```

Example 2:

```
DATAEXTRACT WITH LOCK
     name, city  FROM hotel
     WHERE zip   LIKE '9*'
     ORDER BY name;
OUTFILE hotel.list
```

Example 3:

```
DATAEXTRACT
     customer.cno, name, reservation.arrival, price
     FROM customer, reservation
     WHERE customer.cno = reservation.cno;
OUTFILE cres.data
```

The database query is formulated in the same way as a SELECT statement in SQL, except that the keyword DATAEXTRACT or DATAEXTRACT WITH LOCK is used instead of SELECT. The query must produce an unnamed result table. All options of the SELECT statement are allowed here:

- selecting the result columns and determining their sequence in the result table,
- joining several tables,
- selecting result rows by using qualifications,
- defining a particular sort sequence.

The query must always end with a semi-colon (;).

If the option WITH LOCK is specified, all tables from which rows are to be selected are read-locked so that other users cannot modify these tables during the extract run.

The name of the target file is specified after OUTFILE. Usually, the target file is a disk file. The DATAEXTRACT statement has the effect that the result table is written to the target file. If the target file already exists, it is completely overwritten; otherwise a new one is created.

The data of the target file can be sent directly to the tape device or printer. For testing purposes, some rows of the result table can be displayed on the screen. The filename specific to the operating system (see the "User Manual Unix" or "User Manual Windows") can be used for selection.

If two OUTFILE descriptions are specified, Load generates a DATALOAD statement for the extracted data. The statement will, however, only be executable if only one table was used for data selection and all mandatory columns were included in the SELECT list. As usual, the first filename designates the statement file, the second the data file.

Format Specifications for the Output File

The format specifications related to the file described in 3.4, "Format Specifications Related to a File and Other File Options" DEC (decimal representation), DATE (date representation), TIME (time representation), ASCII or EBCDIC (code conversion), etc., are also allowed in DATAEXTRACT statements. These options may be specified in any order.

For output files, the option APPEND which determines that an existing file with the same name will not be overwritten. The extracted data is written successively to the end of this file instead.

The options COMPRESS, SEPARATOR ’<character>’ and/ or DELIMITER ’<character>’ can be used to produce a compressed output file. The data is written without leading or closing blanks, with each column value separated by a separating character. The default SEPARATOR is the comma. Character strings (not numbers) are enclosed in double quotation marks when the DELIMITER option does not specify something else.

If the output file becomes so large that it must be distributed over several magnetic tapes, then the option COUNT must be used. The number of records that fit on the tape must be specified after COUNT. In this case, Load writes a line with the sequence number of the partial file to each tape. If the specified number of lines was written, the user is requested to mount the next tape.

Structure of the Target File

If no specifications for the structure of the target file have been made, the following rules apply:

- Each result row produces an output record.
- The sequence of the output records is determined by the result table.
- The sequence of columns in the output record is determined by the result table.
- The first column starts at position 1 of the output file.
- All output values are output in plaintext.
- The column values are separated from each other by a character string (may be empty) currently set as separator (see Section User-specific Set Parameters).

Diverging from these default rules, it is possible to specify exactly which columns are to be written to which positions of the output record.

Example 1:

```
DATAEXTRACT title, name, account FROM customer;
     title     01-05
     name      10-19
     account   20-29
OUTFILE  custextract
```

Example 2:

```
DATAEXTRACT customer.cno, name, reservation.rno, hno
     FROM customer, reservation
     WHERE customer.cno = reservation.cno;
     1     01-05
     2     07-13
     3     15-19
     4     21-25
OUTFILE  custextract
```

A column description assigns the values of a result column to a specific output field in the target file.

If column descriptions are used, the query should only select columns that are output or used for sorting in an ORDER BY condition.

The sequence of column descriptions is arbitrary.

The columns can be specified with their names or their sequence numbers in the result table.

The output fields are described by their starting and ending positions, just like the fields for DATALOAD. Gaps between fields are of no consequence because Load fills these gaps with blanks, if necessary.

If an output field is longer than required, numerical values are justified to the right and character strings to the left. The remaining gaps are padded with blanks.

If an output field is shorter than required, character strings are truncated on the right. The execution of the DATAEXTRACT statement will be rejected for numerical values, if the loss of significant digits is to be expected.

Data Formats in the Target File

On request, Load converts the output values into the specified data format (see Section External Data Formats).

Example:

```
DATAEXTRACT * FROM item;
     itno      01-08  CHAR
     descr     09-39
     stock     40-43  INTEGER
     min_stock 44-45  INTEGER
     price     47-53  DECIMAL(2)
     weight    54-57  REAL
OUTFILE ...
```

The external data format need only be specified in an extract statement if the relevant column is to be output in another format than CHAR. Output in CHAR format is possible for all column types.

Load can convert columns of the type FIXED, SMALLINGT, and INTEGER into any of the numerical data formats INTEGER, DECIMAL, ZONED, and REAL; if, in doing so, any significant digits could be lost, a syntax error is reported. Refer to the annotations for the external data format INTEGER in Section External Data Formats.

FLOAT-type columns can only be converted into REAL (or CHAR) ones.

In order to edit the output file in every environment, each of the mentioned data formats can also be supplied in hexadecimal representation with two hexadecimal digits representing one byte. In this case, the data type must be provided with the additional specification HEX, namely

```
[CHAR] HEX                      or
[CHAR] FLOAT HEX                or
INTEGER HEX                     or
REAL HEX                        or
DECIMAL [fraction] HEX          or
ZONED [fraction] HEX
```

Each HEX-formatted data field needs exactly double the space it would need for the same format without the HEX option specification.

Options for the Output of Numerical Columns

All numerical columns of the result table can be edited by using the functions SCALE, ROUND, and TRUNC. The data can have any external format because the functions are applicable to columns that are to be output in CHAR format as well as to columns in REAL, ZONED, or other formats.

The scaling factor specified after the keyword SCALE can be positive or negative. The value to which the function is referring will be multiplied by the corresponding decimal power.

The ROUND and TRUNC functions determine the fractional digits of a number. The number n of fractional digits must lie between 0 and 18. If the number has no fractional digits, the functions have no effect.

TRUNC n means that the n+1st and all the following fractional digits of the number are set to 0, while the first n fractional digits remain unchanged.

ROUND n means that the number is to be rounded from right to left starting with the n+1st fractional digit. If this digit is greater than or equal to 5, the nth digit will be incremented by 1. In this case, too, the result is a number with the n+1st and all the following fractional digits equal to 0; but the first digits of the number may be modified by rounding up.

Both ROUND and TRUNC can be applied in combination with the SCALE function. The functions must be specified in the following order: SCALE before ROUND or TRUNC. The order of processing corresponds to this order.

Example:

```
DATAEXTRACT * FROM distance
     ...
     cm     10-13 INTEGER  SCALE 2
     meter  14-17 INTEGER
     km     18-21 INTEGER  SCALE -3 ROUND 0
OUTFILE dimensions.bin
```

If numbers are output in plaintext, the CHAR FLOAT option can be used to obtain a floating point representation of these numbers, regardless of their size. If the SCALE function is applied and numbers to be output become so large or so small that they cannot be represented any more as fixed point numbers, then they are automatically output in floating-point format.

Output of NULL Values

A character string that can be defined using the SET command is used to represent NULL values in the target file.

For each output column, a particular constant can be declared that will be written into the output record when a NULL value occurs:

Example 1:

```
DATAEXTRACT hno, arrival, departure FROM reservation;
     hno       01-05
     arrival   07-13
     departure 15-21 IF NULL SET POS 15-29 = ’permanent_guest’
OUTFILE  ...
```

The keywords IF and SET can be omitted, whereas the keywords NULL and POS are mandatory. The default operator is '='. Further operators are not allowed.

The position specification of the output field for the constant is made in the same way as all other position specifications. Load does not check whether this output field overlays other fields.

It is only necessary to specify the data format of the constant- in the output file when it is not the default CHAR. In any case, the constant must be specified within the statement as plaintext value enclosed in single quotation marks.

If the constant is to be output in one of the numerical data formats INTEGER, DECIMAL, or ZONED, it must have a valid number format. This means that it must either be a floating point number in mantissa/exponent representation, or a fixed point number in the currently determined or default decimal representation.

Example 2:

```
DATAEXTRACT itno,name,price,expiration_date FROM item;
     itno            01-03
     name            07-26
     price           30-42 ZONED NULL POS 30-42 ZONED '-1,00'
     expiration_date 46-51
OUTFILE  item.extract
    DATE 'yymmdd'
    DEC  '/ /.''
```

Specifying a NULL condition for NOT NULL columns does not produce a syntax error as it does for DATALOAD and DATAUPDATE, because no subsequent errors can result from it.

Text Constants in the Target File

Additional fields with text constants can be placed into the target file among those output fields which are filled from the result table:

Example:

```
DATAEXTRACT cno, firstname, name FROM customer;
     'Customer Number:' 1 - 15
     'Name :'          22 - 28

     cno       16 - 21
     firstname 29 - 38
     name      39 - 48
OUTFILE  ...
```

The text constant enclosed in single quotation marks is specified in this case instead of the column name or column number.

The text constant is output in CHAR format and truncated or padded with blanks on the right, if necessary.

There is no sequence for the description of output fields; output columns and output constants can be mixed.

If the statement contains the constant but no column descriptions, all result columns are entered into the output record in accordance with the default conventions. Constants entered previously may be overwritten in this case.

Generating Command Files with DATAEXTRACT

A special format of the DATAEXTRACT statement can be used if you want to write the data of a table into an external file and make this file be usable as a command file for reloading the table as well.

DATAEXTRACT FOR DATALOAD

Syntax:

```
DATAEXTRACT [WITH LOCK]
     FOR DATALOAD | FASTLOAD
     TABLE <table name>
     [ <order clause> ] ;
<external outfile spec>
[ ; <external outfile spec> ]
```

Example:

```
DATAEXTRACT FOR DATALOAD
     TABLE customer ;
OUTFILE customer.load ;
OUTFILE customer.data
```

This statement generates a command file that allows a table to be completely restored. In contrast to the simple DATAEXTRACT statement, it is therefore not possible to exclude columns or rows from the table.

The command file contains a CREATE TABLE statement only when the table is a base table and belongs to the executing user. The command file always contains a DATALOAD statement and the complete table contents.

When the option WITH LOCK is specified, the table will be read locked during execution so that no simultaneous modifications can be made to it.

When FASTLOAD is specified, a FASTLOAD statement is generated instead of the usual DATALOAD statement. The FASTLOAD statement contains a USAGE clause. 80 is the default percentage. According to the user’s requirements, this percentage can be changed to any value between 50 and 100 by editing the generated command file.

If the order of the table rows in the file is important, e.g., for FASTLOAD, you need to formulate an ORDER BY statement.

As a precaution, table and column names are treated as <special identifier>s in the output file and are enclosed in double quotation marks. This notation is mandatory if a name contains special characters or is identical to an SQL keyword, or if upper- and lowercase characters are to be distinguished.

When two OUTFILE specifications are made, the first file contains the statements, the second one the data. This allows statements to be edited when the table contains BYTE columns or is very large.

File options like ASCII/EBCDIC, DATE format, etc. refer only to the file for which they have been specified. The SET values will be inserted for missing options.

When the COMPRESS option is specified for the generated file of statements, it has the effect that the column names are only output in their actual lengths. When the COMPRESS option is specified for the data, it has the same effect as for a normal DATAEXTRACT statement.

The generated DATALOAD statement contains all the file options so that the user is independent of the current SET statement values for loading.

As the same character string is always used to represent NULL values when extracting data, the generated DATALOAD statement contains DEFAULT NULL conditions for all optional columns. The NULL representation used is recorded as file option NULL ’<string>’.

The selected data is written into the file in accordance with the default conventions. An explicit description of the output format (e.g., INTEGER, SCALE specification) is not possible.

If the table is empty, a special DATALOAD statement is generated that contains a selection condition in the format IF POS n-m < > ’<literal>’.

A line containing this literal at the specified position is written to the desired data file.

Thus a file having the specified name is created with a separate command and data files, even if a table contains no data. This ensures that a load run is not interrupted because Load does not find the specified file.

If the data is written to the file containing the statements, then there is at least one line included for a DATALOAD statement. This guarantees that the next statement of the command file is not misunderstood as data .

No data is entered into the table formerly empty or its counterpart, because the line generated by Load does not meet the selection criterion.

DATAEXTRACT FOR DATAUPDATE

This statement generates a command file which contains a DATAUPDATE statement and the extracted data. This command file can be used for restoring defined contents after modifying the table.

Syntax:

```
DATAEXTRACT [ WITH LOCK ]
     FOR DATAUPDATE TABLE <table name>
     [ <order clause> ] ;
<external outfile spec>
 [ ; <external outfile spec> ]
```

Example:

```
DATAEXTRACT WITH LOCK
     FOR DATAUPDATE TABLE customer ;
OUTFILE customer.upd
```

The table <table name> must have at least one key column. Otherwise, it would be impossible to unambiguously restore the backed up state.

An SQL SELECTstatement is executed. It selects all the columns and all the rows of the table. When the option WITH LOCK is specified, the table is read locked during execution so that no simultaneous modifications can be made to it. When an ORDER BY clause is specified, it is added to the SELECT statement.

A DATAUPDATE statement is generated for the table and written into the specified file. With this statement, modifications made to the table can be rolled back to the state represented by the selected data.

As a precaution, the table and column names are treated as <special identifier>s and are enclosed in double quotation marks. This notation is mandatory if a name contains special characters or is identical with an SQL keyword or if upper- and lowercase characters are to be distinguished.

If two OUTFILE specifications are made, the first file contains the statements, the second one the data.

The DATAUPDATE statement contains DEFAULT NULL conditions for the optional columns of the table. The NULL representation used is recorded as file option NULL ’<string>’.

The DATAUPDATE statement contains complete specifications of all formats and can therefore be executed independently of the current SET specification (see Section DATAEXTRACT FOR DATALOAD).

The selected data is written into the file in accordance with the default conventions. An explicit description of the output format (e.g., DECIMAL, IF-NULL-SET-POS) is not possible.

Empty tables are handled in the same way as in case of DATAEXTRACT FOR DATALOAD (see Section DATAEXTRACT FOR DATALOAD).

Test Output on Screen

Entering OUTFILE * displays a section of the result table on the screen.

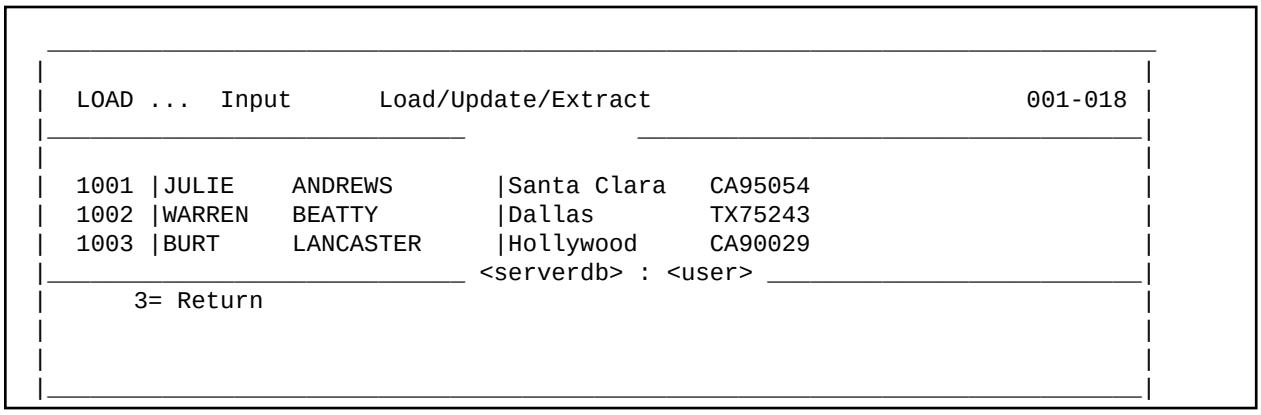
If the default representation is chosen, this section shows the left upper part of the result table. Beginning with the first row, only as many rows are displayed as fit a screen page, and rows longer than the screen width are truncated on the right.

Explicit column descriptions must be formulated so that the output rows fit the screen; otherwise, an error is reported.

The displayed line area can be shifted using a RUN option (see Section RUN with Range Option) to determine a starting row that is other than the first row of the result table.

Load displays the number of output rows after the return to the screen.

Example:


LOAD ... Input Load/Update/Extract 001-018
1001 |JULIE ANDREWS |Santa Clara CA95054
1002 |WARREN BEATTY |Dallas TX75243
1003 |BURT LANCASTER |Hollywood CA90029
<serverdb> : <user>
3= Return